04

Estados Update en la entidad

Transcripción

[00:00] Hola. En el video anterior nosotros hablamos sobre los ciclos de vida de una entidad, hablamos también sobre los estados de estas entidades al movernos dentro de este ciclo de vida. Creamos el DAO para la entidad categoría, que son los métodos que nos permiten realizar operaciones de guardar, actualizar o remover en la base de datos y creamos el primer método dentro de ese DAO que era el método guardar.

[00:27] Nosotros en la clase anterior vimos este diagrama donde teníamos que cuando nosotros instanciamos una entidad, ella se encuentra en un estado transiente, que es un estado inicial donde la entidad no es considerada para ser guardada dentro de la base de datos.

[00:47] Luego, cuando nosotros realizamos un persist, usando el EntityManager, esa entidad pasa a estar con un estado Managed o un estado gerenciado. Ya a partir de este punto ya esa entidad es considerada para ser guardada dentro de la base de datos.

[01:01] Entonces, JPA va a prestar atención en esa entidad y en los valores que tenga para enviarlos o sincronizarlos en la base de datos y realiza un commit o un flush. Esos valores son enviados a la base de datos o sincronizados donde se crea un id y un nuevo registro.

[01:17] Luego al realizar un close o un clear, esa entidad pasa a estar en un estado detached, que es un estado separado, similar al estado transiente donde

ya no es más considerada para ser guardada o no va a considerar las nuevas alteraciones que se realizan en esta entidad.

[01:36] Ahora en esta parte yo voy a hablar sobre el método merge, que es un método que nos permite traer esas entidades que se encuentran como detached al estado managed nuevamente, para poder guardar nuevas alteraciones en la base de datos.

[01:52] Entonces con eso, nosotros vamos a crear el método actualizar dentro de nuestro DAO, que nos va a permitir actualizar nuevos valores dentro de registros ya existentes en la base de datos.

[02:04] Nuevamente voy a hacer una aclaración de los métodos commit y flush y el método close y clear. La diferencia entre el método commit y flush es que el método flush nos permite realizar un rollback en las operaciones, es decir si nosotros realizamos algún error en las operaciones nosotros podemos devolvernos a los valores iniciales.

[02:25] Con el commit, los valores que sean enviados a la base de datos son definitivos. De igual forma con el método close y clear, cuando yo realizo el método close, yo cierro el EntityManager y tendría que instanciar nuevamente un EntityManager para poder realizar operaciones dentro de este ciclo de vida.

[02:46] Ahora cuando yo realizo el clear del EntityManager yo simplemente envío todas las entidades a un estado detached para ahorrar espacio en memoria en el proyecto.

[02:57] De vuelta en mi clase registro de producto, lo voy a reemplazar esa clase commit em.flush. Y para refrescar lo que habíamos visto anteriormente, voy a ejecutar esta aplicación recordando que yo había instanciado una entidad de la entidad categoría, pasó a un valor celular, realicé una actualización de ese parámetro luego de haber hecho la persistencia de celulares a libros.

[03:30 Cerré la conexión, ese método lo vamos a reemplazar por el clear, pasando la entidad celulares a un estado detached y voy a realizar una nueva actualización de ese valor, modificar ese valor de “LIBROS” a “SOFTWARE” para ver qué ocurre.

[03:50] Entonces voy a ejecutar esta aplicación, voy a guardar. Está cargando. Y ahora en la consola él me muestra que creó las tablas, como hemos visto anteriormente, realizó el insert de la categoría celulares, realizó la modificación, el update a “LIBROS” pero no realizó una segunda actualización a “SOFTWARE”.

[04:18] Entonces, según lo que habíamos mencionado, yo tengo que llamar el EntityManager y utilizar el método merge y me solicita una entidad. Esa entidad va a ser la entidad celulares, que es la que ya instancié previamente que se encuentra con un estado detached.

[04:35] Entonces vamos a ver qué pasa si de este modo, él ya pasa esa entidad de un estado detached a un estado Managed y guarda ese nuevo valor en la base de datos. Entonces aquí yo tendría que agregar un EntityManager y voy a realizar otro flush, para sincronizar los valores en la base de datos.

[04:58] Entonces, voy a ejecutar la aplicación, guardo los valores y en la consola se muestra que me está mostrando un error, donde se me está diciendo que no existe un constructor default para la entidad. ¿Entonces, qué es lo que está pasando?

[05:17] Cuando yo uso el método merge, JPA necesita realizar un select dentro de la base de datos para poder realizar modificaciones en el proyecto. Entonces yo voy a ir a la clase producto, voy a agregar mi constructora default, generar constructor usando campo, no le voy a enviar ningún campo y voy a generar.

[05:45] Y voy a guardar. Y vamos a hacer lo mismo en la clase categoría. Voy a generar un constructor default, que es un constructor sin parámetros. Generar

constructor usando los campos. Y generar. Entonces, ya con eso tengo los constructores para categoría y para producto.

[06:07] Vamos a ejecutar la aplicación nuevamente, guardamos los cambios y ahora sí, vemos que él está creando las tablas, él realizó un insert, realizó un update, referente al campo para libro, pero no realizó un segundo update.

[06:24] Él realizó un select. ¿Entonces qué es lo que está ocurriendo aquí? Entonces está seleccionando ese registro en la base de datos y lo está reasignando en mi proyecto con un nuevo registro de memoria.

[06:39] Entonces, para poder yo realizar una alteración en el proyecto, en el registro anterior de la entidad categoría, lo llamamos celular y yo tengo que reasignar ese registro que estoy llamando en memoria a ese valor antiguo. Entonces lo que voy a hacer es resignar celulares a ese EntityManager merge.

[07:01] Entonces en teoría, luego que yo haga esa reasignación, cuando ya ejecute mi aplicación él va a realizar el select y en la misma secuencia va a realizar el update para ese valor de celulares.

[07:14] Vamos a ejecutar esta aplicación, guardamos los cambios y vemos que efectivamente creó las tablas, él realizó el insert cuando se creó la entidad, realizó la actualización cuando pasamos al libro, vamos a ver mostrar aquí de nuevo, realizó el update, realizó el select cuando hizo el merge y trajo los valores de la tabla en la base de datos.

[07:39] Y modificó de “LIBROS” a “SOFTWARE” cuando nosotros asignamos en el setter el valor de “SOFTWARE”. Enviamos ese valor, lo sincronizamos con el método flush y con eso quedó actualizada nuestra identidad categoría. Ahora solamente nos está faltando actualizar el DAO.

[08:02] Entonces vamos a ir a la clase CategoriaDao, que se encuentra en el proyecto inicial y aquí voy a crear un nuevo método. Ese método va a ser del

tipo público, no va a retornar nada y se va a llamar actualizar. Va a recibir un elemento del tipo categoría y lo vamos a llamar categoría.

[08:30] Entonces, dentro de ese método, vamos a utilizar el EntityManager, this.em y vamos a usar el método merge, al cual vamos a pasar como parámetro esa categoría. Cerramos y ya con esto tenemos el segundo método de la clase DAO. En la siguiente clase voy a explicar sobre el método delete, que nos va a permitir remover un registro de la base de datos.